Modifying WAVEWATCH III ® code

Hendrik L. Tolman
Chief, Marine Modeling and Analysis Branch
NOAA / NWS / NCEP / EMC

Hendrik.Tolman@NOAA.gov

Overview

- How does compiling of WAVEWATCH III work.
  - Why not direct FORTRAN code?
  - How does preprocessor work.
- Adding a pre-compile model option.
- Adding a subroutine.
- Internal data structure.
- Best practices.
  - Programming style
  - Adding to the model.
  - Manual and documentation.
  - Subversion repository.
  - Testing.

Compile basics

- WAVEWATCH III is intended as both
  - a general modeling framework, and
  - an efficient operational wave model.

- This implies that
  - it should be possible to include lots of options in the modeling framework, but
  - that the final code as compiled should include only that what is needed.

- To achieve this, the WAVEWATCH III source code is not plain FORTRAN 90, but needs to go through a preprocessor to obtain the FORTRAN 90 code.

Manual chapter 5

Compile basics

- WAVEWATCH III follows the FORTRAN 90 standard, and all files are build as complete modules to enable and enforce interface checking and use association.

- A typical WAVEWATCH III file is named ID*name*md.ext

ID	Type identifier	w3	Basic wave model routine
		wm	Multi-grid extension routine
		ww3_	Main program
		gx_	GrADS postprocessor.
name	Code ID, typically 4 characters, e.g. “srce”		
md	Identifier that this is a module.		
ext	File extension	ftn	Code requiring preprocessing
		f90	Plain FORTRAN 90 code.

Compile basics

- For instance,

  w3srcemd.ftn contains the module of the basic wave model that processes source terms, and that will need to be preprocessed before it can be compiled.

  wmwavemd.ftn contains the multi-grid wave model module, requiring preprocessing.

  ww3_grid.ftn contains the main program ***ww3_grid*** and requires preprocessing.

  mod_xnl4v5.f90 does not follow the convention, except that the file extension indicates that the file does not need to be preprocessed.

    This is part of G. Van Veldder exact interactions package that is distributed with the wave model.

Compile basics

- Below is part of w3srcemd.ftn, where input source terms are computed.
- The lines starting with !/XXX are optional pieces of code, activated by their “switches” XXX, in this case for

  linear input, or

  exponential input.

```
!
! 2.  Calculate source terms ------------------------------------- *
!
! 2.a Input.
!
!/LN1        CALL W3SLN1 (       WN1, FHIGH, USTAR, U10DIR , VSLN       )
!/LNX        CALL W3SLNX
!
!/ST1        CALL W3SIN1 ( SPEC, WN2, USTAR, U10DIR ,        VSIN, VDIN )
!/ST2        CALL W3SIN2 ( SPEC, CG1, WN2, U10ABS, U10DIR, CD, Z0,        &
!/ST2                                                  FPI, VSIN, VDIN )
!/ST3        CALL W3SIN3 ( SPEC, CG1, WN2, U10ABS, USTAR, DAIR/DWAT, AS,  &
!/ST3                 U10DIR, Z0, CD, TAUWX, TAUWY, VSIN, VDIN, LLWS )
!/STX        CALL W3SINX
!
```

Compile basics

- Switches to be using in the compilation are stored in the switch file:

  This file is stored as ./bin/switch in the main WAVEWATCH III directory. The installation script makes links to the original file in most work directories.

  If the switches NL1 and ST2 are present in the switch file, the corresponding part of the preprocessed code w3srcemd.f90 will become :

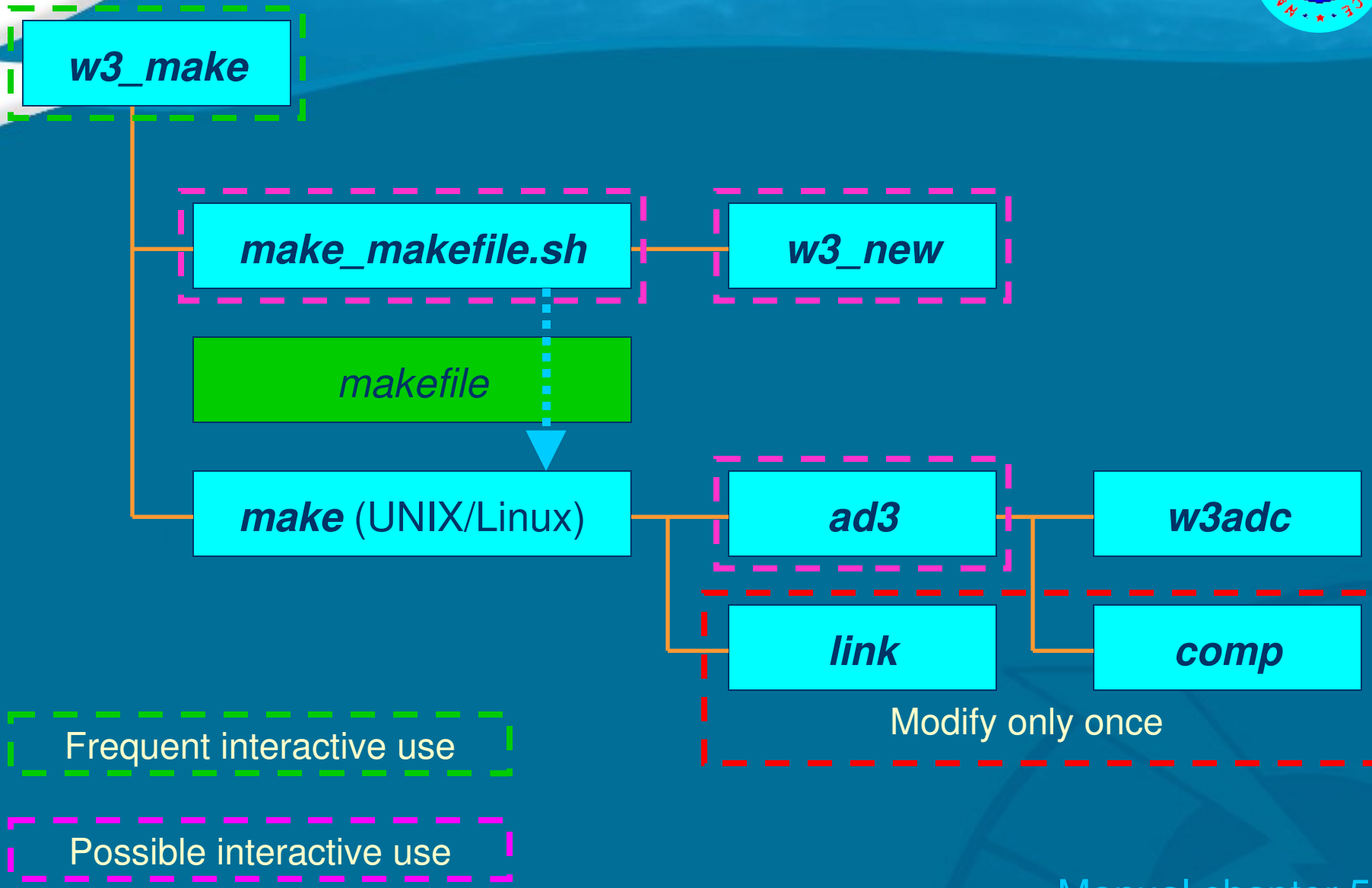
```
!
! 2.  Calculate source terms ------------------------------------- *
!
! 2.a Input.
!
       CALL W3SLN1 (       WN1, FHIGH, USTAR, U10DIR , VSLN       )
!
       CALL W3SIN2 ( SPEC, CG1, WN2, U10ABS, U10DIR, CD, Z0,        &
                                                  FPI, VSIN, VDIN )
```

How does this work

- A FORTRAN 77 program w3adc.f is compiled during installation of the model to produce the program ***w3adc***.
- ***w3adc*** is managed by the script ***ad3***, also put in place during model installation.
- ***ad3*** also uses the ***comp*** script, in which compiler options are set.
- ***ad3*** and the ***link*** script are called in the makefile, which is used by the standard UNIX/Linux ***make*** facility
- The makefile is updated by the script ***make_makefile.sh***, every time the switch file is modified.
- ***make_makefile.sh*** calls ***w3_new*** to touch the appropriate files to be recompiled by ***make***.
- And all this is managed by the ***w3_make*** script.

Sounds complicated **?**

Compile basics



Manual chapter 5

In summary:

- Only ***w3_make*** is normally used :

  w3_make ww3_grid compiles this program only.

  w3_make by itself compiles all recognized WAVEWATCH III programs.

- ***ad3*** can be run interactively, particularly if test output needs to be switched on in selected routines.
- ***make_makefile.sh*** and ***w3_new*** can be run interactively as indicated in the manual.
- The rest of the system you will never see after the model is installed, but …..

- It is essential that ***comp*** and ***link*** are set up with compiler error capturing if codes are to be edited.

Manual chapter 5

Does and don’ts

- Even if the system may look a little complicated, do use it by properly modifying the ***.ftn*** files.

  This is the only way of modifying this inside WAVEWATCH III in such a way that it can be ported to the distribution version of the model.

  It is therefore more or less required by the license.

- ***w3_source*** will give you the clean FORTRAN files and the corresponding makefile:

  Use this for operational implementations of the model.

  Don’t use this for upgrading source code.

  Use it for MS Windows applications ….

  …. better still, don’t use Windows ….

Modifying code

Code may need to be updated for bug fixes, or as part of systematic model development.

- For simple edits my preferred way to work is:

  Use ***ln3*** to make a link to the file to be updated in the ./work directory under the main wave model directory.

  Edit the link in the ./work directory, and test there with ***w3_make*** and by running standard tests.

  Note: there is a link to the switch file in this directory to modify the model configuration.

  After the modification is satisfactory, remove the links from the ./work directory.

- HINT:: use ***arc_wwatch3*** to make archive files before and after code modification, if no other management tool like subversion is used. The resulting ***.tar*** files can be re-installed with ***install_wwatch3***.

Manual section 5.5

Modifying code

- If systematic modifications or additions to the code are made, there will likely be a need for:
  - Adding subroutines in existing modules.
  - Adding subroutines in new modules.
  - Adding old switches to existing subroutines.
  - Adding new switches to the model.

- These actions will be discussed in the following slides, and are also described in section 5.5 of the manual.
- Note that if a new module with new switches is included, instructions for both modifications need to be followed.

- See HINT on previous slide …..

Manual section 5.5

Adding subroutines in existing modules.

- This is in principle simple. Add the code and recompile using ***w3_make***.

- A complication may occur is the subroutine is used by other modules. In that case, the proper “use” statement needs to be added to the calling module.

  This may modify relations between files in the makefile and ***make*** commands.

  Run ***make_makefile.sh*** manually to assure that the makefile is updated, before ***w3_make*** is run.

  This only needs to be done if “use” statements are modified.

Manual section 5.5

Adding subroutines in new modules.

- This typically adds a new file like w3coolmd.ftn or mypackage.f90 to the model files.
- To assure that the new files are included in the compilation, ***make_makefile.sh*** needs to be modified as follows:

  Add module name to sections 2.b and 2.c to assure inclusion in the makefile under proper conditions.

  Add module name and object file names to section 3.b to assure proper dependencies in makefile.

  Run ***make_makefile.sh*** manually and check makefile in ./ftn directory for proper inclusion of new file.

- NOTE: ***make_makefile.sh*** checks use statements in .f90 (preprocessed) files to determine file dependencies.

Manual section 5.5

Adding old switches to existing subroutines.

- Relationships of switches to model files are maintained in the ***w3_new*** script.
- If old switches are added to new files the following actions are needed:

  Add the new file to the lists of files to be touched in section 2 of ***w3_new***.

  If the switches include use statements, interactively run ***make_makefile.sh*** to assure that the makefile is updated as needed.

Manual section 5.5

Adding new switches to the model.

- After a new switch is added to an existing file, the following action is required.

  If the switch is part of a new group of switches of which one is to be selected, add a new ‘keyword’ ($key) to section 2 of ***w3_new***.

  Update files to be touched in section 2 of ***w3_new*** as necessary.

  Add ‘keyword’ and/or switches to section 2 of ***make_makefile.sh***.

  Run ***make_makefile.sh*** and check consistency of ./ftn/makefile.

Manual section 5.5

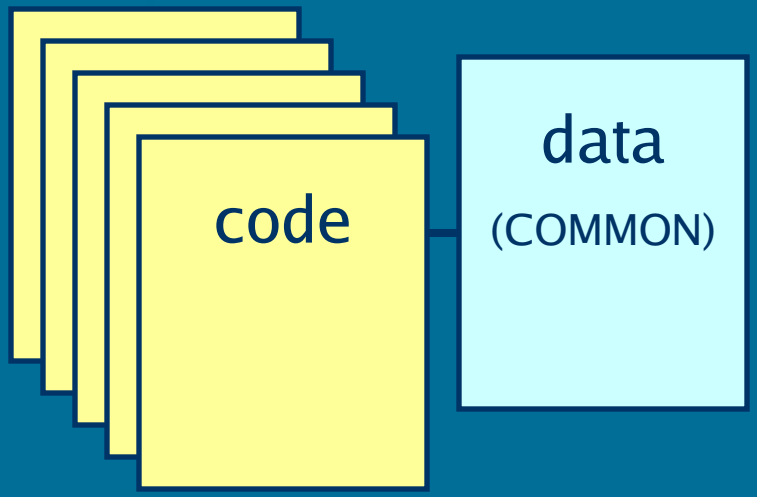
When adding to the wave model, it is essential to understand how data is stored.

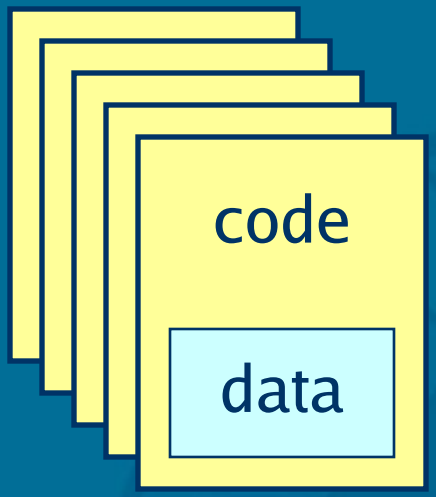
Model version 1.18 (1999)

- FORTRAN 77
- COMMON data structure
- Single static data structure.

Model version 2.22 (2002)

- FORTRAN 90
- Modular
- Object oriented, static data structure bundled with code

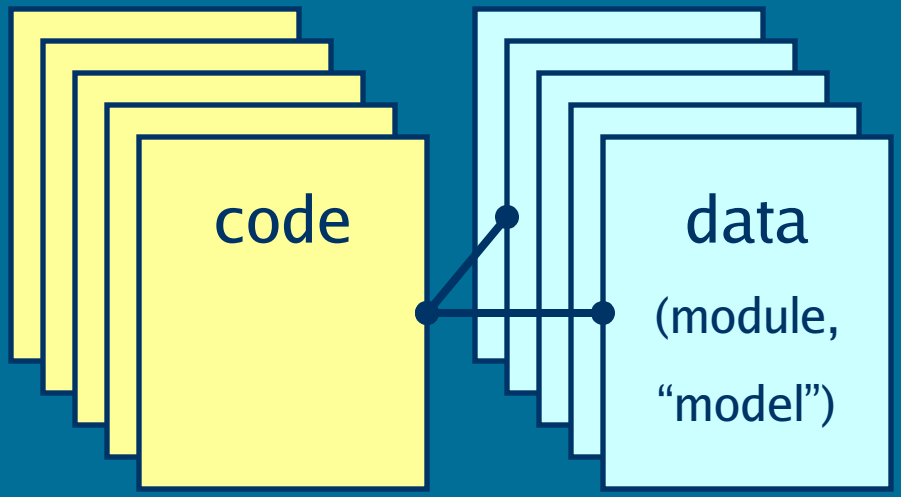




Data structure

Model version 3.06 (2005)

- Modular FORTRAN 90
- Dynamic / multiple data structure (modular)
- Small overhead (7% on Linux, 2% on IBM SP)



Present status :

- F77 and COMMON data structures are obsolete.
  - Exception are aux codes like w3adc.f.
- Data embedded in modules largely obsolete.
  - Use in model development, see best practices.
- Data in data modules now the norm in 3.14.
  - Exception: file constants.ftn.

Data structures

How is this done?

- Inside the code variables look like they are defined for a single grid, for instance, the grid dimensions `NX`, `NY`, and a bottom depth array `ZB`.
- However, these variables are declared as pointers.
- The actual data is stored in a user-defined type `GRID`.
- An array of `GRIDS` of this type allows for data of multiple grids to be stored simultaneously.
- The pointers are then set to represent values of the grid currently under consideration.

```
!/
!/ data structure
!/
      TYPE GRID
        INTEGER        :: NX, NY
        REAL, POINTER  :: ZB(:,:)
      END TYPE GRID
!/
!/ Data storage
!/
      TYPE(GRID), TARGET,                &
                  ALLOCATABLE :: GRIDS(:)
!/
!/ Pointers
!/
      INTEGER, POINTER :: NX, NY
      REAL, POINTER    :: ZB(:,:)

!/ ************************************

      NX => GRIDS(I)%NX
      NY => GRIDS(I)%NY
      ZB => GRIDS(I)%ZB
```

Data structures

- There are many data structures defined in the model.
- All essential model data for model setup as well as dynamic wave conditions is stored in five data modules:

w3gdatmd.ftn	Grid and model setup data.
w3adatmd.ftn	Auxiliary data used and stored internal to the model only.
w3idatmd.ftn	Model input data.
w3wdatmd.ftn	Basic wave model state.
w3odatmd.ftn	Model output data.

- Each module contains data for as much grids as identified in the mosaic (including model input and spectral point output grids).

Manual sections 6.5 & 6.6

Best practices

- For those who want to modify / contribute to WAVEWATCH III, a best practices guide is available.
- Note that as a part of the license, additions made to the model have to be offered to NCEP for inclusion in future model distributions.
- Best practices cover :
  - Programming style
  - Adding to the model.
  - Manual and documentation.
  - Subversion repository.
  - Regression testing.
- These issue will be touched upon briefly here, but the guide will be the authoritative source.

Best practices guide

Programming style:

- Use WAVEWATCH III documentation style (see templates).
- Use coding style:

  Free format but layout as in old fixed format.

  Upper case for permanent code, lower case for temporarily code. Latter can be included as permanent testing using `!/Tn` switches.
- Maintain update log at header f documentation.
- Embed all subroutines in modules or main programs, using naming convention outlined before.
- Follow FORTRAN 90 standard, with best practices as outlined in section 2 of the guide.
- Provide appropriate documentation in LaTeX format for inclusion in the manual.

Best practices

subroutine template

```
!/ ------------------------------------------------------------------ /
      SUBROUTINE W3XXXX
!/
!/                  +-----------------------------------+
!/                  | WAVEWATCH III           NOAA/NCEP |
!/                  |             John Doe              |
!/                  |                          FORTRAN 90 |
!/                  | Last update :         01-Jan-2010 |
!/                  +-----------------------------------+
!/
!/    01-Jan-2010 : Origination.                        ( version 4.xx )
!/
!  1. Purpose :
!  2. Method :
!  3. Parameters :
!
!     Parameter list
!     ------------------------------------------------------------------
!     ------------------------------------
!
!  4. Subroutines used :
!
!      Name      Type  Module   Description
!     ------------------------------------
!      STRACE    Subr. W3SERVMD Subroutine
!     ------------------------------------
!
!  5. Called by :
!
!      Name      Type  Module   Description
!     ------------------------------------
!     ------------------------------------
!
!  6. Error messages :
!  7. Remarks
!  8. Structure :
!  9. Switches :
!
!     !/S  Enable subroutine tracing.
!
! 10. Source code :
!
!/ ------------------------------------------------------------------ /
!/S      USE W3SERVMD, ONLY: STRACE
!/
      IMPLICIT NONE
!/
!/ ------------------------------------------------------------------ /
!/ Parameter list
!/
!/ ------------------------------------------------------------------ /
!/ Local parameters
!/
!/S      INTEGER, SAVE           :: IENT = 0
!/
!/ ------------------------------------------------------------------ /
!/
!/S      CALL STRACE (IENT, 'W3XXXX')
....
!/
!/ End of W3XXXX ---------------------------------------------------- /
!/
      END SUBROUTINE INSBTX
```

Best practices

module template

```
!/ ------------------------------------------------------------------ /
      MODULE W3XXXXMD
!/                  +-----------------------------------+
!/                  | WAVEWATCH III           NOAA/NCEP |
!/                  |             John Doe              |
!/                  |                          FORTRAN 90 |
!/                  | Last update :         01-Jan-2010 |
!/                  +-----------------------------------+
!/
!/    01-Jan-2010 : Origination.                        ( version 4.xx )
!/
!/    Copyright 2010 National Weather Service (NWS),
!/       National Oceanic and Atmospheric Administration.  All rights
!/       reserved.  WAVEWATCH III is a trademark of the NWS.
!/       No unauthorized use without permission.
!/
!  1. Purpose :
!  2. Variables and types :
!
!      Name      Type  Scope    Description
!     ------------------------------------
!     ------------------------------------
!
!  3. Subroutines and functions :
!
!      Name      Type  Scope    Description
!     ------------------------------------
!      W3XXXX    Subr. Public   ........
!     ------------------------------------
!
!  4. Subroutines and functions used :
!
!      Name      Type  Module   Description
!     ------------------------------------
!      STRACE    Subr. W3SERVMD Subroutine
!     ------------------------------------
!
!  5. Remarks :
!
!
!  6. Switches :
!
!     !/S  Enable subroutine tracing.
!
!  7. Source code :
!/
!/ ------------------------------------------------------------------ /
!/
      PRIVATE
!/
      CONTAINS
!/ ------------------------------------------------------------------ /
      SUBROUTINE W3XXXX
.....
!/
!/ End of W3XXXX ---------------------------------------------------- /
!/
      END SUBROUTINE W3XXXX
!/
!/ End of module W3XXXXMD ------------------------------------------- /
!/
      END MODULE W3XXXXMD
```

Programming style cont'ed:

- If existing packages are added to the wave model, then such packages do not need to be re-coded to conform to our standards.
- Such packages will require interface routines, which are expected to confirm to the standards.
- Copyright of NWS may extend to interface routines, but obviously not to linked in packages.

Adding to the model (no NCEP subversion access)

- Propagation schemes and source terms:
  - Use available “user-defined” dummy modules.
  - Follow coding guidelines.
  - Provide file with necessary modifications to w3srcemd.ftn, w3wavemd.ftn, and all other model files that need to be updated.
  - Provide (previous) test cases with expected results.
  - Make each module self-contained.
    - Define all variables in the module header. We will integrate them in the full data structure.
    - Separate initialization and computation as outlined in the dummy modules.

Adding to the model (no NCEP subversion access)

- For more intricate modifications to the code, consult with NCEP code managers on how to do this and on how to provide this to NCEP.
- New pre- or postprocessors should be provided in their entirety, included in the compile and link system.
- HINT: when developing new source terms, include and test them in the postprocessors ***ww3_outp*** and ***gx_outp*** first, before including/testing them in actual wave model integration.

Adding to the model (with NCEP subversion access)

- Same rules apply as for those without svn access with following exceptions:
  - NCEP code managers will assign switches to new sources and propagation scheme to be used instead of the ‘X’ switches.
  - Developers will be responsible for integration in the data structure:
    - Do this only after rigorous testing of self-contained system.
  - NCEP code managers will add new code to the TRUNK of the repository. Changes to be provided relative to most recent TRUNK, not to most recent distributed version

Manual and documentation.

- Provide full LaTeX documentation for inclusion in the manual:
  - NCEP svn users have access to manual, and are expected to add to it directly.
    - NCEP will provide editing.
  - Others provide separate files.
    - NCEP will integrate.
  - Use BibTEX.
  - Use dynamic references to equations, figures and tables only.

Subversion repository.

- See description in the guide.

  Full documentation in GNU ChangeLog format required.

Testing

- Regression testing procedures will be established in the near future, most likely based on experiences and tools provided by Erick Rogers and Tim Campbell from NRL Stennis.

The end

End of lecture wwws 3.3